An Approach to a Test Oracle for XML Query Testing

Dae S. Kim-Park, Claudio de la Riva, Javier Tuya

University of Oviedo
Computing Department
Campus of Viesques, s/n, 33204 (SPAIN)
kim_park@lsi.uniovi.es, claudio@uniovi.es, tuya@uniovi.es

Abstract. XML queries are broadly used in Web environments, but the existing approaches towards software quality based on testing have not deeply addressed them. Although there are some works oriented to generate test inputs for testing XML queries, the evaluation of expected outputs against the actual outputs resulting from the tests has not been tackled as far as we are concerned. In this paper, a research proposal is presented to deal with the absence of the expected outputs when testing XML queries, focusing the efforts on the definition of a feasible test oracle.

1 Introduction and motivation

In recent years, Web Engineering has emerged as a new discipline motivated by the swift growth of the World Wide Web. This growth has been accompanied by an increasing complexity of Web applications (WebApps) as we can clearly see, for example, in today's WebApps formed by compositions of Web Services created, in turn, with heterogeneous fast-evolving technologies. The development of these complex WebApps, like other software systems, should involve quality criteria to guarantee a certain degree of reliability based on testing, verification and validation activities. Regarding testing, current approaches on WebApps are commonly centered in checking static and dynamic navigational paths [9][15][11]. However, much functionality of WebApps depends on data access operations, for example, to retrieve and manipulate data requested by the user or by a software component in execution. For this reason, it is expected that testing data access operations may have some impact on the improvement of the quality of WebApps.

There are well-known technologies intended to carry out data access operations, such as the SQL language for relational databases, but XML-based technologies for data querying are becoming popular as is the use of XML-based formats for data representation and interchange in the Web. The existing approaches and tools to test data access operations on XML-based applications are commonly conceived to test data repositories or query engines [2][1], while the queries utilized to retrieve the data (XML queries) are generally ignored, although being prone to faults [7]. In previous works [7][4][5], we entailed this concern by defining a technique to automatically generate the inputs for testing XML queries. However, the expected outputs for the

tests were dismissed despite they are a key factor to determine the correctness of test executions.

To address the lack of the expected outputs, some researchers state that there should be an entity, called *oracle*, capable to determine whether or not a program under test has behaved as expected during execution. The definition of such oracle derives in the so called *oracle problem*, as oracles are usually difficult or even impossible to obtain [8]. In order to overcome this problem, it is assumed that a human can act as an oracle by manually checking the correct behaviour of the program under test, but this approach may be unfeasible in some circumstances. Particularly, a human oracle for XML query testing raise some problems because (1) the test inputs may be large, (2) there may be many queries to test, and (3) there may be numerous test cases to consider.

In this research proposal we present a line of work intended to tackle some of the problems concerning testing of queries for data access in XML-enabled environments (such as the Web), with a focus on the definition of a test oracle oriented to solve the absence of the expected outputs during testing.

The paper is structured as follows. In section 2 the aims and objectives of the research are outlined. Section 3 briefly describes the current work and, finally, in section 4 the expected contributions of the research and plans for future work are presented.

2 Research hypotheses and objectives

The current research activity in the field of test oracles is not significant in contrast to the huge variety of the existing approaches involving test input generation. Amongst the works encompassing test oracles, partial oracles seem to be the best alternative to XML query testing. Such partial oracles are capable to determine whether an actual output of a test case is incorrect while ignoring the correct output [8]. Thus, they do not need to infer the expected output of a test in order to provide a diagnostic about the correctness of the target program. In XML query testing, this is a remarkable advantage considering the large volumes of XML data that may be provided as test inputs and/or outputs.

With these considerations, our research hypothesis is intended to prove that a partial oracle is a viable alternative when dealing with large volumes of data for testing. In particular, we are aiming to prove that a partial oracle may be suitable to test data access operations based on queries in XML-enabled environments. To cover the hypothesis, the following objectives are in scope:

1. Define a feasible partial oracle to test XML queries. The automation of the oracle will be one of the main points to consider.
2. Validate the proposal to prove its effectiveness in a real testing scenario, which may include developing a proof of the concept to show the practical application of the proposal.

3. Find possible alternatives to the proposed oracle if required. The proposed oracle may not be suitable in every case. Because of that, other alternatives should be studied, specially the existing ones that could be adapted to XML query testing, such as metamorphic testing [3].

3 Proposed approach

The complexity of the XML queries is a concern since there are many broadly used alternatives oriented to query XML data. XML queries are commonly expressed using the XPath [12] and XQuery [13] languages, but also other programming technologies can be used to cover the similar functionalities, such as the Java language supported by the SAX/DOM API [10][14]. To overcome this heterogeneity, we abstracted the representation of the queries under test by treating them as black-box programs, each of which receives an XML document as input, and outputs an XML document fragment (a set of XML structures not necessarily well-formed) resulting from the query operation. Formally, let $q : D \rightarrow F$ be the query under test, where D is the set of all possible inputs (XML documents), and F is the set of all possible outputs (XML document fragments). Then, q is a black-box query program under test that takes an XML document $I \in D$ and produces the actual output $q(I) \in F$.

With this abstraction, we propose the use of a partial oracle to test the query programs according to a specification composed by the following two elements:

- Behavioural requirements, which establish a specification about the correct behaviour of a determined target query program under test. This specification is loose, which means that it need not be complete, and
- Oracle constraints, which define invariant properties of the expected outputs that must be satisfied by every correct query program. Each oracle constraint is intended to check the presence a known type of fault.

Since behavioural requirements depend on the target program, the tester must provide them manually. In contrast, oracle constraints are invariant; thus, they could be embedded in the partial oracle internals and their evaluation could be automated.

Fig. 1 shows the structure of the oracle and its integration in the testing environment. As it can be seen, the oracle receives as input the behavioural requirements, as well as the test input and the actual output of the test, and yields a "Pass" or "Fail" response as a result of the oracle constraints evaluation.

Because oracle constraints specify general properties of the expected outputs, prior to their evaluation, they are particularized to the target programs under test by means of the behavioural requirements. Whenever an oracle constraint is not satisfied given a set of behavioural requirements, it is understood that the partial oracle has detected a fault, in which case, the partial oracle should respond with a "Fail" message.

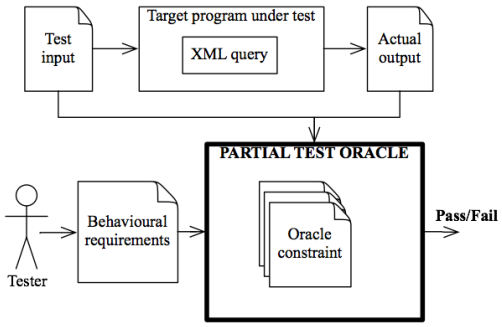

Fig. 1. Partial test oracle in the testing environment.

In the paper in [6] we detail the partial oracle approach with a set of behavioural requirements and oracle constraints, including a case study. Some of the proposed behavioural requirements and oracle constraints are presented in the next sections.

3.1 Behavioural requirements

In order to ease the manual specification of the query program under test, behavioural requirements are loose, which means that they do not comprise a complete specification of the target program. However, the precision of the partial oracle will increase as the tester specifies these requirements closer to the complete specification. This provides flexibility to balance the relationship between the cost and the precision of the oracle.

The behavioural requirements are given by the tuple

$$S = < A_s, D_s, t_s >,$$

where:

- A_s is a set of XML nodes (XML elements or attributes) that are expected to be in the actual output of the target program, but are not included in the test input,
- D_s is a set of XML nodes that are expected to be in the input, but not in the actual output.
- t_s is a relaxed specification-compliant query. It is defined as a function with the same domain and image as the query under test q. Then, the property $q_s(I) \subseteq t_s(I)$ must hold for the definition of t_s, where q_s denote the correct implementation of the query that meets the specification, also in the domain and image of q, and $q_s(I)$ is the expected output for the test input I. Note that q_s it is not available for the tester. In practice, t_s is expected to be a less expressive query than q_s, and hence, it should be less error-prone and easier to specify. For example, suppose that we need to obtain the title of an XHTML Web page only if its body contains the string value `"Contents"`. The specification-compliant query, q_s would be represented with the XPath syntax as `/html/head/title[../body = "Contents"]`. Then, t_s could be

defined as the query `/html/head/title`, which always returns the title of the page, independently of the body contents. As seen, t_s should be simpler and easier to code than q_s.

The elements of the behavioural requirements tuple need not be defined as a complete specification. It is not required to specify every possible node in A_s or D_s, and the query t_s does not need to be as expressive as q_s in order to detect some types of errors. In the next section, the use of these behavioural requirements is shown.

3.2 Oracle constraints

Oracle constraints set necessary conditions for the correct behaviour of the queries under test. Since these constraints are intended to deal with operational aspects of the querying processes, they are domain-independent, and thus, can be embedded in the oracle (as shown in Fig. 1). Some of the oracle constraints we currently propose are presented and described below.

- $q(I) \bigcap A_s = A_s$. This constraint establishes that the XML nodes specified in the set A_s must be contained in the actual output, $q(I)$.
- $q(I) \bigcap D_s = \emptyset$. The constraint establishes that the actual output must not contain nodes from D_s.
- $q(I) \subseteq t_s(I)$. It checks that the actual output is contained in the relaxed specification-compliant query. If this constraint is satisfied, it means that data selection faults (such as mistaken predicates or bad node references) were not detected in the query under test, q.

Note that each constraint makes use of a different behavioural requirement from the tuple in Section 3.1, and every behavioural requirement could be defined with a custom precision by the tester.

After a test execution, if any of the constraints does not hold for a given query program, then a fault has been detected, and the oracle could notify the cause of that fault with a human-readable message for the tester.

4 Expected contributions and plan for future work

The expected contribution of the research to Web Engineering consists on improving the quality of WebApps by means of testing XML queries. For this purpose, we focus on an approach to a test oracle whose main characteristics are outlined below:

1. Allows the tester to balance the cost and the effectiveness of the oracle. The tester can provide a loose specification (behavioural requirements) of the program under test, instead of giving a complete specification, which would be more costly and complex.
2. The proposed oracle is highly automatable. The mechanism of the oracle depends on the rules defined by oracle constraints whose evaluation could be automated.

At this moment, the proposed oracle can detect a relatively small set of faults. For future work, we plan to add/modify behavioural requirements and oracle constraints (as shown in the example in Section 3) to enhance the fault detection capabilities of the oracle. It is also necessary to define a systematic method for the evaluation of oracle constraints as this will result in the automation of the oracle. Besides, the oracle proposal should be validated to prove its effectiveness. We are planning to do the validation by using experimental techniques such as mutation, and by providing real/industrial case studies.

5 Acknowledgements

This work was partially funded by the Department of Education and Science (Spain) and ERDF funds within the National Program for Research, Development and Innovation, project Test4SOA (TIN2007-67843-C06-01) and the RePRIS Software Testing Network (TIN2007-30391-E).

6 References

[1] D. Barbosa, A.O. Mendelzon, "Declarative Generation of Synthetic XML Data", Software Practice and Experience, vol 36, pp. 1051-1079, 2006.
[2] A. Bertolino, J. Gao, J. Marchetti, A. Polini, "Automatic Test Data Generation for XML Schema-based Partition Testing", Proceedings of Automation of Software Test, pp. 4-11, 2007.
[3] T.Y. Chen, F.-C. Kuo, T.H. Tse, Z.Q. Zhou, "Metamorphic Testing and Beyond", Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice, 00: 94:100, 2003.
[4] D.S. Kim-Park, C. de la Riva, J. Tuya, J. García-Fanjul, "Generating Input Documents for Testing XML Queries with ToXgene", Proc. of the 3rd IEEE Testing: Academic and Industrial Conference, Fast Abstract Track, 2008.
[5] D.S. Kim-Park, C. de la Riva, J. Tuya, J. García-Fanjul, "Synthetic Data Generation for XML Query-aware Testing", 6th Workshop on System Testing and Validation, 2008.
[6] D.S. Kim-Park, C. de la Riva, J. Tuya, "A Partial Test Oracle for XML Testing", Testing: Academic and Industrial Conference – Practice and Research Techniques, 2009.
[7] C. de la Riva, J. García-Fanjul, J. Tuya, "A Partition-Based Approach for XPath Testing", Proceedings of the International Conference on Software Engineering Advances, Washington, DC, USA, 2006.
[8] E.J. Weyuker, "On Testing Non-testable Programs", The Computer Journal, 25(4): 465-470, 1982.
[9] F. Ricca, P. Tonella, "Analysis and Testing of Web Applications", Proceedings of the 23rd International Conference on Software Engineering, 2001.
[10] SAX Project, http://www.saxproject.org/, 2009.
[11] S. Elbaum, S. Karre, G. Rothermel, "Improving web application testing with user session data", Proceedings of the 25th International Conference on Software Engineering, 2003.
[12] World Wide Web Consortium, "XML path language 2.0 (XPath 2.0)", http://www.w3.org/TR/xpath20/, 2007.

[13] World Wide Web Consortium, "XQuery 1.0. An XML query language", http://www.w3.org/TR/xquery/, 2007.
[14] World Wide Web Consortium, "Document Object Model (DOM)", http://www.w3.org/DOM/, 2009.
[15] Y. Deng, P Frankl, J. Wang, "Testing web database applications", SIGSOFT Software Engineering Notes, 29(5), 2004.